QUICK REVIEW

1. A function that does not have a data type is called a void function.
2. A return statement without any value can be used in a void function. If a return statement is used in a void function, it is typically used to exit the function early.
3. The heading of a void function starts with the word void.
4. In C++, void is a reserved word.
5. A void function may or may not have parameters.
6. A call to a void function is a stand-alone statement.
7. To call a void function, you use the function name together with the actual parameters in a stand-alone statement.
8. There are two types of formal parameters: value parameters and reference parameters.
9. A value parameter receives a copy of its corresponding actual parameter.
10. A reference parameter receives the address (memory location) of its corresponding actual parameter.
11. The corresponding actual parameter of a value parameter is an expression, a variable, or a constant value.
12. A constant value cannot be passed to a reference parameter.
13. The corresponding actual parameter of a reference parameter must be a variable.
14. When you include & after the data type of a formal parameter, the formal parameter becomes a reference parameter.
15. The stream variables should be passed by reference to a function.
16. If a formal parameter needs to change the value of an actual parameter, in the function heading, you must declare this formal parameter as a reference parameter.
17. The scope of an identifier refers to those parts of the program where it is accessible.
18. Variables declared within a function (or block) are called local variables.
19. Variables declared outside of every function definition (and block) are called global variables.
20. The scope of a function name is the same as the scope of an identifier declared outside of any block.
21. See the scope rules in this chapter (section, Scope of an Identifier).
22. C++ does not allow the nesting of function definitions.
23. An automatic variable is a variable for which memory is allocated on function (or block) entry and deallocated on function (or block) exit.
24. A static variable is a variable for which memory remains allocated throughout the execution of the program.
25. By default, global variables are static variables.
26. In C++, a function can be overloaded.
27. Two functions are said to have different formal parameter lists if both functions have:

    • A different number of formal parameters or

    • If the number of formal parameters is the same, then the data type of the formal parameters, in the order you list them, must differ in at least one position.

28. The signature of a function consists of the function name and its formal parameter list. Two functions have different signatures if they have either different names or different formal parameter lists.
29. If a function is overloaded, then in a call to that function, the signature— that is, the formal parameter list of the function—determines which function to execute.
30. C++ allows functions to have default parameters.
31. If you do not specify the value of a default parameter, the default value is used for that parameter.
32. All of the default parameters must be the far-right parameters of the function.
33. Suppose a function has more than one default parameter. In a function call, if a value to a default parameter is not specified, then you must omit all arguments to its right.
34. Default values can be constants, global variables, or function calls.
35. The calling function has the option of specifying a value other than the default for any default parameter.
36. You cannot assign a constant value as a default value to a reference parameter.